A Hierarchical Environment for Virtual Supercomputing

New Haven, Connecticut
(203) 777-7442
Software@LindaSpaces.com
http://www.LindaSpaces.com
A Hierarchical Environment for Virtual Supercomputing

New Ideas

- Cost-effective HPC platforms will employ hierarchical architectures with heterogeneous components
- Virtual shared memory systems can provide the conceptual foundation for a uniform, high-performance software environment

Impact

- Portable, high-performance software for parallel machines
- Reduction in cost of software development and maintenance
- Increased software lifetimes
Long-Range Hierarchical Target Architecture

Cluster of hierarchical memory machines

Multiple SMPs connected on high performance network fabrics

Individual SMPs
Objectives

- Investigate API design & implementation issues for parallel/distributed systems with hierarchical memory
- Provide code portability among parallel architectures including SMPs, distributed memory machines, & clusters
- Achieve high performance through
  - Language-level processing and optimization
  - Exploitation of native compilers and libraries
  - Intelligent dynamic process/data mapping
- Achieve ease of use by providing a uniform logical software architecture across the entire hierarchy
Initial Target Hardware Architecture

Nodes: Shared Memory Multiprocessors (SMPs) (either PCs or workstations)

Interconnect: Fast Ethernet or a switch fabric (e.g., Myrinet, Giganet, Quadrics)

Operating Systems: Windows NT, Linux, Vendor UNIX
Possible Approaches for Linda System Software

- **Network Linda**
  - Treat each node of each SMP as a separate, full-fledged “machine” on the network

- **Simple Hybrid System**
  - Paradise server for VSM
  - Native SMP or Original Linda on each SMP

- **Hierarchical System**
  - One VSM server per SMP, with all linked together to manage a single VSM
  - Intelligent run-time system to exploit locality through redirection, caching and data relocation.
Hierarchical Linda Software Architecture

Linda Process

- In Proxy
- In Proxy Thread
- Process Memory
- Thread Create
- Proxy Manager Thread
- Application Thread
- Eval Server Thread
System Design I

Hierarchical Linda System
Threads of Control

Eval Server

in global eval count array
Update global eval count array
out global eval count array
Spawn local user TOC

Proxy Manager

Receive Message
Store tuple in local shared memory
Waiters?
No
Wake appropriate waiters
Yes
Spawn local service TOC
Main thread continues
New handler thread

Beginning of new TOC
Check for match
Match?
No
Register as waiter
Yes
Send tuple to remote TOC
Exit

Wake up
Sleep
Linda Code Segments Embedded in User TOCs

**in/rd Processing**
- Part of current user TOC
- Check for match
  - Yes: Wake up
  - No: Match?
    - Yes: Get tuple from local shared memory
    - No: Send in/rd request to remote manager
- Send in/rd request to remote manager
- Receive tuple from remote handler
- Register as waiter
- Sleep
- Return

**eval Processing**
- Part of current user TOC
- Send out request and tuple to remote manager
- Store tuple in local shared memory
- Waiters?
  - Yes: Wake appropriate waiters
  - No: Out eval request tuple for selected processor
- Return
- Select processor for eval and update global count array

**out Processing**
- Part of current user TOC
- Local Set?
  - Yes: Send out request and tuple to remote manager
  - No: Wake appropriate waiters
- Return